

Distributed Asynchronous Dual Free Stochastic Dual Coordinate Ascent

Zhouyuan Huo
zhouyuan.huo@mavs.uta.edu

Heng Huang
heng@uta.edu

November 22, 2016

Abstract

With the boom of data, it is challenging to optimize a large-scale machine learning problem efficiently. Both of the convergence rate and scalability are nontrivial for an optimization method. Stochastic gradient descent (SGD) is widely applied in large-scale optimization, and it sacrifices its convergence rate for a lower computation complexity per iteration. Recently, variance reduction technique is proposed to accelerate the convergence of stochastic method, for example stochastic variance reduced gradient (SVRG), stochastic dual coordinate ascent (SDCA) or dual free stochastic dual coordinate ascent (dfSDCA). However, serial algorithm is not applicable when the data can not be stored in a single computer. In this paper, we propose Distributed Asynchronous Dual Free Coordinate Ascent method (dis-dfSDCA), and prove that it has linear convergence rate when the problem is convex and smooth. The stale gradient update is common in asynchronous method, and the effect of staleness on the performance of our method is also analyzed in the paper. We conduct experiments on large-scale data from LIBSVM on Amazon Web Services and experimental results demonstrate our analysis.

1 Introduction

With the proliferation of information, the size of data grows larger, and the optimization of large-scale machine learning model becomes a challenging problem. To handle large-scale data, stochastic gradient descent and its variants are proposed for its low computation complexity per iteration [1]. However, when dataset can not be stored and processed in a single machine, we have to use distributed optimization method. There are many works on distributed machine learning systems like Parameter Server [11] or GraphLab [14]. Traditional serial optimization methods are extended to distributed version for large-scale problem, such as stochastic gradient descent and its variants (SGD) [12, 25, 10, 28, 17, 6], stochastic coordinate descent (SCD) [13], alternating direction method of multipliers (ADMM) [24], stochastic dual coordinate ascent (SDCA) [23, 8, 21]. There are mainly two architectures in distributed system: one is shared-memory architecture, and the other is distributed-memory architecture. Distributed-memory architecture is about the distributed system with multiple machines, in this paper, we consider distributed-memory architecture with one server and multiple workers. All the communication happens between server and each worker. There are mainly two types of communication, synchronous communication and asynchronous communication. In synchronous method, server has to wait and get information from all workers before it goes to the next step. It is known that the synchronous method suffers from straggler problem. To alleviate the communication overhead, asynchronous methods are proposed. In these methods, server can update parameters with stale information. Although the stale gradient information introduces noise in the process, it does not necessarily affect the convergence property of asynchronous method. There are also some research to alleviate this noise by using staleness variable in the update rule [27, 15]. Throughout this paper, we only consider asynchronous communication between each worker and server.

Because of the variance of stochastic gradient introduced by SGD method, a decreasing learning rate has to be applied to guarantee its convergence. Thus, it leads to a sub-linear convergence rate for this kind of method. In strongly convex problem, the convergence rate is $O(1/T)$ if we use SGD method. Recently, variance reduction technique is proposed, and SVRG [9], SAGA [3], proximal SVRG [22], SDCA [20] and Dual Free SDCA (dfSDCA) [19] are proved to have linear convergence rate when problem is strongly convex. SDCA is one of the most popular methods to solve ℓ_2 regularized problem [20, 9], and it is proved to enjoy strong theoretical convergence guarantee property. It can also be viewed as a variation of SGD method with unbiased estimate of gradient. As its name indicates, SDCA is optimizing a dual problem derived from primal problem. However, the dual problem is likely to be meaningless sometimes. In [19], a variation of SDCA (dfSDCA) is proposed and applied to problem in which its dual problem is unavailable. In spite of the improvement on convergence property, there is no work to use dfSDCA in distributed system.

In the paper, we propose a Distributed Asynchronous Dual Free Coordinate Ascent (dis-dfSDCA) method. We also analyze the convergence property of our method and prove that it has linear convergence when the problem is convex. We perform experiments on the large-scale data from LIBSVM, and the results prove that our proposed method outperforms the state-of-the-art distributed SDCA and demonstrate our theoretical analysis.

2 Related Work

In this paper, we consider following ℓ_2 -norm regularized empirical loss minimization problem:

$$\min_{w \in \mathbb{R}^d} P(w) = \frac{1}{n} \sum_{i=1}^n \phi_i(w) + \frac{\lambda}{2} \|w\|^2. \quad (1)$$

It is the formulation of many machine learning problems, if $\phi_i(w)$ is hinge loss or logistic loss, it is a classification problem, if $\phi_i(w)$ is least square loss, it is commonly used in regression.

2.1 Stochastic Dual Coordinate Ascent

Instead of optimizing the primal problem (1) directly, some methods are proposed to solve its dual problem:

$$\max_{\alpha \in \mathbb{R}^n} D(\alpha) = \frac{1}{n} \sum_{i=1}^n -\phi_i^*(-\alpha_i) - \frac{\lambda}{2} \left\| \frac{1}{\lambda n} \sum_{i=1}^n \alpha_i x_i \right\|^2 \quad (2)$$

where ϕ_i^* is the conjugate function of ϕ_i , namely, $\phi_i^*(y) = \max_x (x^T y - \phi_i(x))$. In iteration t , firstly we find the optimal solution $\Delta\alpha_i$ to the following subproblem by setting other $\alpha_{j, j \neq i}$ fixed:

$$\max_{\Delta\alpha_i} -\frac{1}{n} \phi_i^*(-(\alpha_i^{t-1} + \Delta\alpha_i)) - \frac{\lambda}{2} \|w^{t-1} + \frac{1}{\lambda n} \Delta\alpha_i x_i\|^2 \quad (3)$$

Then, α_i^t and w^t is updated through:

$$\alpha_i^t = \alpha_i^{t-1} + \Delta\alpha_i \quad (4)$$

$$w^t = w^{t-1} + \frac{1}{\lambda n} \Delta\alpha_i x_i \quad (5)$$

Compared with SGD based method, we do not need to tune learning rate. It is claimed in [5] that SDCA outperforms SGD method in some cases. The analysis in [20] proves that SDCA is a class of method that enjoys the same fast convergence rate as SVRG and its variants. In [9], SDCA method is also illustrated as a

variance reduction method. Different from these SGD based methods, SDCA is able to monitor convergence through computing the duality gap ($P(w(\alpha)) - D(\alpha)$) in the process, where $w(\alpha) = \frac{1}{\lambda n} \sum_{i=1}^n \alpha_i x_i$ denotes the primal dual relation. In [26, 16], importance sampling and adaptive probability sampling strategies are proposed to improve the convergence rate of SDCA. In spite of all these benefits of SDCA method, dual problem is not always available. We can not use SDCA when its dual problem is meaningless.

2.2 Dual Free Stochastic Dual Coordinate Ascent

To solve this problem, [19] proposes dfSDCA method which has the same convergence property as SDCA. In iteration t , w , and α are updated via:

$$\alpha_i^t = \alpha_i^{t-1} - \eta \lambda n (\nabla \phi_i(x_i^T w^{t-1}) + \alpha_i^{t-1}) \quad (6)$$

$$w^t = w^{t-1} - \eta (\nabla \phi_i(x_i^T w^{t-1}) + \alpha_i^t) \quad (7)$$

where $\nabla \phi_i(x_i^T w^{t-1})$ denotes the gradient of ϕ_i with respect to $(x_i^T w^{t-1})$. Although we keep vector $\alpha \in \mathbb{R}^n$ in the optimization, the derivation of dual problem is not necessary for this method. Same as SDCA method, dfSDCA is also an instance of variance reduced SGD, the variance of $(\nabla \phi_i(x_i^T w^{t-1}) + \alpha_i^{t-1})$ goes to zero when t goes to infinity. Besides, according to the update rule in (6) and (7), the primal dual relation also holds for dfSDCA. Authors in [4] improve the convergence rate of Dual Free SDCA by applying non-uniform sampling strategy in each iteration.

2.3 Distributed Optimization

When data can not be stored or processed in a single machine, distributed optimization algorithm will take over the large-scale optimization task. Parameter server framework [11] is developed for large-scale distributed machine learning problems. There are one or more servers, and multiple workers in a cluster. In this paper, we mainly focus on the cluster which has parameter server framework, there is one server storing global parameters, and dataset and workloads are distributed over workers. There are many distributed optimization algorithms proposed based on this frameworks. Distributed SGD is the most popular algorithm, and it is widely used in deep learning optimization [2, 12]. Distributed SVRG is proved to have faster convergence rate than the vanilla distributed SGD on both distributed-memory and shared-memory system. It is proved to have linear convergence rate for strongly convex problem [25, 28, 18], and sub-linear convergence rate $O(1/T)$ for non-convex problem [6]. Distributed SDCA method is also proposed in [7, 21, 23], and CoCoA+ is pointed out to be the state-of-the-art framework. Applying dfSDCA to distributed optimization and analyzing its convergence property are still missing.

3 Distributed Asynchronous Dual Free Stochastic Dual Coordinate Ascent

In this section, we illustrate the details of our algorithm Distributed Asynchronous Dual Free Coordinate Ascent (dis-dfSDCA). Pseudocode of dis-dfSDCA for worker node and server node are presented in Algorithm 1 and Algorithm 2 respectively.

3.1 Update α in Worker

Algorithm 1 presents the pseudocode of dis-dfSDCA in arbitrary worker node k . We assume that a dataset has n samples, and it is evenly distributed over K workers, in each worker k there are n_k samples. $\alpha \in \mathbb{R}^{n_k}$ denotes the dual vector for the data subset in worker k . In the algorithm, H denotes the number of inner

iteration and γ denotes the scaling constant for the update of adaptive probabilities for sample distribution. Initially, α is set to be a zero vector.

Algorithm 1 dis-dfSDCA (Worker k)

Initialize $\alpha \in \mathbb{R}^{n_k}$, $i \in \{1, \dots, n_k\}$, H, γ

repeat

Receive $w^{t-\tau}$ from server;

$v_k \leftarrow 0$;

for $h = 1, 2, \dots, H$ **do**

Option 1: Randomly select sample i from $\{1, \dots, n_k\}$;

Option 2: Select sample i according to adapted probability distribution p , then $p_i \leftarrow \frac{p_i}{\gamma}$;

Update duality gap: $\kappa_i \leftarrow \nabla \phi_i(x_i^T w^{t-\tau}) + \alpha_i^{t-1}$;

Update dual vector: $\alpha_i^t \leftarrow \alpha_i^{t-1} - \eta \lambda n \kappa_i$;

Update update information: $v_k \leftarrow v_k + \kappa_i x_i$;

end for

Send v_k to server;

until Termination

In each worker, we compute the duality gap for each sample $\kappa_i = \nabla \phi_i(x_i^T w) + \alpha_i$, and then update α_i and v_k separately via:

$$\alpha_i^t = \alpha_i^{t-1} - \eta \lambda n \kappa_i, \quad i \in \{1, \dots, H\} \quad (8)$$

$$v_k = \sum_{i=1}^H \kappa_i x_i \quad (9)$$

It is easy to know that when we have optimal solution α^* and w^* , κ_i would be 0, and $\alpha_i^* = -\nabla \phi_i(x_i^T w^*)$. In each inner iteration, we need to select a sample for update. As per the sampling strategy in [19, 26, 4, 16], there are mainly three options: uniform sampling, importance sampling, and adaptive probabilities sampling. Both of importance sampling and adaptive probabilities sampling are proved to have faster convergence rate than uniform sampling. In this paper, we only consider uniform sampling and adaptive probabilities sampling. In the uniform sampling strategy, we select samples randomly or each sample has the same probability to be selected. In the adaptive probabilities sampling strategy, each sample is selected according to the adaptive probability distribution p , where the probability vector $p \in \mathbb{R}^{n_k}$ is coherent with dual residual $\kappa \in \mathbb{R}^{n_k}$. According to [4], the probability distribution p is updated via:

$$p_i(\kappa) = \frac{\sqrt{x_i^T x_i \lambda L + n_k \lambda^2 |\kappa_i|}}{\sum_i \sqrt{x_i^T x_i \lambda L + n_k \lambda^2 |\kappa_i|}}. \quad (10)$$

It is time consuming to compute the probability p in each iteration, so we usually compute p after each pass of all training data. In each iteration, if sample i is selected, we update $p_i = \frac{p_i}{\gamma}$, where constant γ works as a scaling parameter. This step is reasonable, because if one sample is selected in iteration t , its information is used for optimization, there should be lower probability to select this sample in the following iterations. After H iterations, we would send update vector v_k to the server.

3.2 Update w in Server

Algorithm 2 summarizes the pseudocode in server node. The most recent parameter w is maintained in the server, and v_k represents the vector received from worker k in each iteration. Initially, w is set to be a zero

vector.

Because our proposed method uses asynchronous communication, stale parameter will be used in each worker to compute update information. Thus update from slow workers will be received much less often than those from faster workers. Δ is the maximum time delay in our distributed system, which means that update from every worker has to be received at least once by the server every Δ iterations. When $\Delta = 1$, our asynchronous algorithm becomes a synchronous algorithm, and in each iteration, the server has to wait information from all the workers. As in [24], server node keeps τ_k as the time delay of parameter in worker k . Every iteration, we use Φ as the set of workers we have received when the condition $\max(\tau_1, \tau_2, \dots, \tau_K) < \Delta$ satisfies. Then parameter w is updated via:

$$w = w - \eta \sum_{k \in \Phi} v_k \quad (11)$$

After updating τ_k and w , we broadcast the newest w to workers in Φ . From the update rule in our algorithm, it is easy to know that the well-known primal dual relation always holds:

$$w = \frac{1}{\lambda n} \sum_{i=1}^n \alpha_i x_i \quad (12)$$

Algorithm 2 dis-dfSDCA (Server)

Initialize $w \in \mathbb{R}^d$, η , $\tau_i = 1, i = 1, 2, \dots, K$.

repeat

Set $\Phi = \emptyset$;

repeat

Receive v_k from worker k ;

Add k to set Φ ;

$\tau_k = 0$;

until $\max(\tau_1, \tau_2, \dots, \tau_K) < \Delta$;

$z \leftarrow 0$.

for worker $k \in \Phi$ **do**

$\tau_k \leftarrow 1$;

$z \leftarrow z + v_k$;

end for

for worker $k \notin \Phi$ **do**

$\tau_k \leftarrow \tau_k + 1$;

end for

Update $w^t \leftarrow w^{t-1} - \eta z$;

Broadcast w to workers in Φ ;

until Termination

4 Convergence Analysis

In this section, we provide the convergence rate analysis of our method. We only consider the theoretical analysis of dis-dfSDCA with Option 1, and $H = 1$ in worker side. Thus, in each iteration, the update rule in a worker node can be written as,

$$\alpha_i^t = \alpha_i^{t-1} - \eta \lambda n (\nabla \phi_i(x_i^T w^{t-\tau}) + \alpha_i^{t-1}) \quad (13)$$

$$v_i^t = (\nabla \phi_i(x_i^T w^{t-\tau}) + \alpha_i^{t-1}) x_i \quad (14)$$

where $w^{t-\tau}$ denotes stale parameter, which is τ older than the current parameter. The update rule in the server node is as follows:

$$w^t = w^{t-1} - \eta v_k \quad (15)$$

For further analysis, we make the following assumptions for problem (1). All of them are common assumptions in the theoretical analysis for distributed stochastic gradient method.

Assumption 1 $\nabla\phi_i$ is Lipschitz continuous, thus inequality holds that:

$$\|\nabla\phi_i(x) - \nabla\phi_i(y)\|_2 \leq L\|x - y\|_2. \quad (16)$$

Assumption 2 ϕ_i is convex, it is true that:

$$\phi_i(x) \geq \phi_i(y) + \nabla\phi_i(y)^T(x - y). \quad (17)$$

Assumption 3 Maximum of time delay τ is smaller than Δ .

We split the update of our algorithm into two stages, the inner loop is a traversal of all data, namely $t \in \{1, \dots, n\}$, and the outer loop $s \in \{1, \dots, S\}$ denotes the number of epoch. We define the message transmitted between worker and server $v_i^{s,t} = \left(\nabla\phi_i(x_i^T w^{s,t-\tau}) + \alpha_i^{s,t-1} \right) x_i$, and we have $\mathbb{E}[v_i^{s,t}] = \nabla P(w^{t-\tau})$.

Lemma 1 If ϕ_i is convex and $\nabla\phi_i$ is Lipschitz continuous, let w^* be the optimal solution to problem (1) and $\alpha_i^* = -\nabla\phi_i(x_i^T w^*)$. For any stale parameter $w^{t-\tau}$:

$$\begin{aligned} \sum_{t=1}^n \mathbb{E} \|\nabla\phi_i(x_i^T w^{t-\tau}) x_i + \alpha_i^* x_i\|^2 &\leq 2L^2 \eta^2 \Delta^2 \sum_{t=1}^n \mathbb{E} \|v_i^t\|^2 \\ + 4L \sum_{t=1}^n \left(P(w^{t-1}) - P(w^*) - \frac{\lambda}{2} \mathbb{E} \|w^{t-1} - w^*\|^2 \right) \end{aligned} \quad (18)$$

Following Lemma 1, we are able to analyze the convergence property of our method.

Theorem 1 If ϕ_i is convex and $\nabla\phi_i$ is Lipschitz continuous, and let w^* be the minimizer of $P(w)$, and $\alpha_i^* = -\nabla\phi_i(x_i^T w^*)$. If $\eta \leq \frac{1}{2L+n\lambda+4L\Delta+2\Delta^2L}$, then we have:

$$\begin{aligned} &\left[\|w^{s,0} - w^*\|^2 + \frac{1}{2n\lambda L} \sum_{i=1}^n \|\alpha_i^{s,0} x_i - \alpha_i^* x_i\|^2 \right] \\ &\leq e^{-\eta\lambda s} \left[\|w^{0,0} - w^*\|^2 + \frac{1}{2n\lambda L} \sum_{i=1}^n \|\alpha_i^{0,0} x_i - \alpha_i^* x_i\|^2 \right]. \end{aligned} \quad (19)$$

Then we analyze the convergence of objective function value.

Theorem 2 From Theorem 1, let $\eta = \frac{1}{2L+n\lambda+4L\Delta+2\Delta^2L}$ it follows that whenever:

$$T \geq O \left(\left(\frac{2L + 4L\Delta + 2\Delta^2L}{\lambda} + n \right) \log \left(\frac{1}{\varepsilon} \right) \right) \quad (20)$$

the inequality holds:

$$P(w^{s,0}) - P(w^*) \leq \varepsilon \quad (21)$$

From Theorem 1 and Theorem 2, we know that our dis-dfSDCA has linear convergence for convex optimization. From [19], dis-dfSDCA is also an instance of variance reduced SGD method. Right now, we can obtain the upper bound of the variance of v_i^t . Because $\mathbb{E}\|v_i^t - \nabla P(w^{t-\tau})\|^2 = \mathbb{E}\|v_i^t\|^2 - \|\nabla P(w^{t-\tau})\|^2$, all three terms are greater or equal than zero, so we analyze the upper bound of $\mathbb{E}\|v_i^t\|^2$ here.

Corollary 1 *Let w^* be the optimal solution to problem (1) and $\alpha_i^* = -\nabla\phi_i(x_i^T w^*)$, as t goes to infinity, $\mathbb{E}\|v_i^t\|^2$ goes to zero:*

$$\begin{aligned} \mathbb{E}\|v_i^t\|^2 &= \mathbb{E}\|\alpha_i^{t-1}x_i - \alpha_i^*x_i + \alpha_i^*x_i + \nabla\phi_i(x_i^T w^{t-\tau})x_i\|^2 \\ &\leq 2\mathbb{E}\|\alpha_i^{t-1}x_i - \alpha_i^*x_i\|^2 + 2\mathbb{E}\|\alpha_i^{t-1}x_i + \nabla\phi_i(x_i^T w^{t-\tau})x_i\|^2 \\ &\leq 2\mathbb{E}\|\alpha_i^{t-1}x_i - \alpha_i^*x_i\|^2 + 2L^2\mathbb{E}\|w^{t-\tau} - w^*\|^2 \end{aligned} \tag{22}$$

where the second inequality follows from the Lipschitz continuity of $\nabla\phi_i$. From Theorem 1, both of $\mathbb{E}\|\alpha_i^{t-1}x_i - \alpha_i^*x_i\|^2$ and $\mathbb{E}\|w^{t-\tau} - w^*\|^2$ go to zero linearly. Therefore, $\mathbb{E}\|v_i^t\|^2$ equals zero finally.

5 Experiments

In this section, we present experimental results on three large-scale real datasets, and evaluate the empirical performance of the proposed optimization algorithm. Our algorithm is implemented in C++, and the point to point communication is handled by openMPI¹. We use Armadillo library² for efficient matrix computation. We run our experiments on Amazon Web Services, and each node is a t2.medium instance which has 2 virtual CPUs.

We use quadratic loss $\phi_i(w) = \frac{1}{2}(w^T x_i - y_i)^2$, with ℓ_2 regularization term $\frac{\lambda}{2}\|w\|^2$, where $\lambda = 0.1$ in all experiments. The experiments are performed on three large datasets from LIBSVM³: IJCNN1, COVTYPE and RCV1. Table 1 shows the details of each dataset.

Dataset	n	d	Sparsity
IJCNN1	49,990	22	41 %
COVTYPE	581,012	54	22 %
RCV1	677,399	47,236	0.16%

Table 1: Experimental datasets from LIBSVM. n denotes the number of data, and d denotes the dimension of feature.

5.1 Convergence of Duality Gap

We compare our method with CoCoA+ [7], which is the state-of-the-art distributed primal-dual optimization framework. CoCoA+ uses synchronous communication, in each outer iteration, the server will not be ready to work until all the information from workers are received. Therefore, it may face straggler problem in the process. This experiment is performed on a cluster with 1 server node and 8 worker nodes. We reimplement CoCoA+ framework using C++, and use SDCA as its local solver for subproblems. Parameters are selected to have the best performance following [7], e.g the iteration number in local solver is the total number of data in each worker. For our method, learning rate is selected from $\eta = \{1, 0.1, 0.001, 0.0001\}$,

¹<https://www.open-mpi.org/>

²<http://arma.sourceforge.net/>

³<https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html>

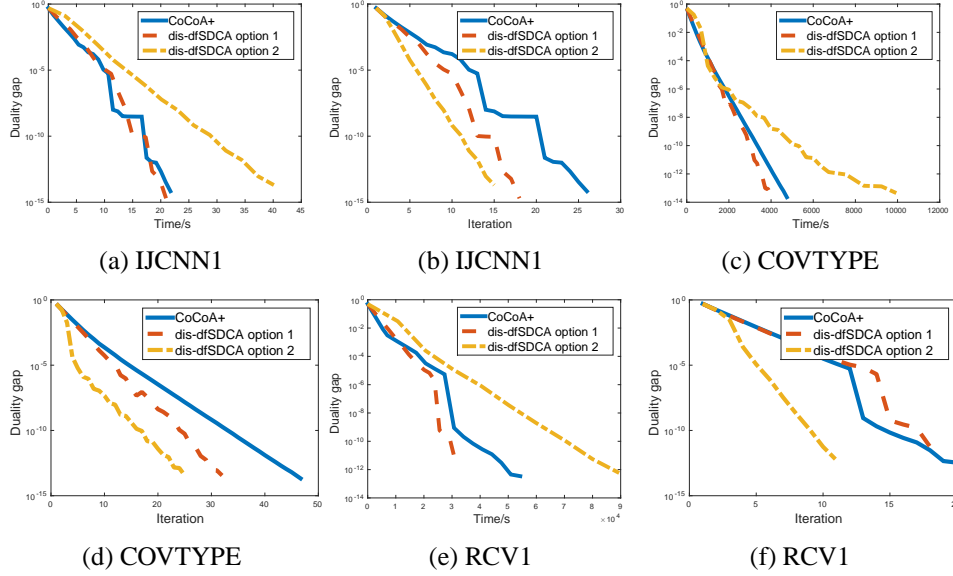


Figure 1: Duality gap convergence of three compared methods. The first column shows the convergence of duality gap with respect to time, and the second column shows the performance of each method with respect to iteration number.

and iteration number in each worker $H = 1000$. We compare the duality gap with respect to time and iteration respectively, where duality gap is well defined in [20].

Figure 1 presents the performance of compared methods on IJCNN1, COVTYPE and RCV1 datasets. First column shows the duality gap with respect to time, we can know that our dis-dfSDCA with Option 1 runs as fast as CoCoA+, and has comparable or even better performance. Option 2 is a little slower than other two methods because in each epoch, it has to compute the distribution probability for data samples, which is very time consuming. In the second column, we show the performance of each method with respect to iteration number, it is obvious that our dis-dfSDCA method with Option 1 converges faster than CoCoA+. As for our dis-dfSDCA method with Option 2, it has the fastest convergence rate because of its adaptive sampling strategy. Therefore, we can draw the conclusion that, our method achieves better performance for large-scale optimization.

5.2 Convergence of $\|\nabla P\|^2$

It is known that $\|\nabla P\|^2 \rightarrow 0$ when $w \rightarrow w^*$. Therefore, we also compare the convergence of $\|\nabla P\|^2$ for three methods. Figure 2 presents the convergence of $\|\nabla P\|^2$ with respect to time and iteration respectively.

From the first column in Figure 2, we find that our dis-dfSDCA with Option 1 runs faster than CoCoA+ method, and dis-dfSDCA method with Option 2 is slower than others because its extra computation of probability distribution. Figures in column two show that our method has faster convergence rate than CoCoA+, Option 2 converges a little faster than Option 1.

5.3 Impact of Worker Number K

In this section, we evaluate the scaling ability of our dis-dfSDCA algorithm. In the experiment, we perform experiments on IJCNN1 and COVTYPE datasets using dis-dfSDCA method with Option 1. Parameters are selected similarly to the parameters in the last two sections. We plot the time speedup in Figure 4, and it is

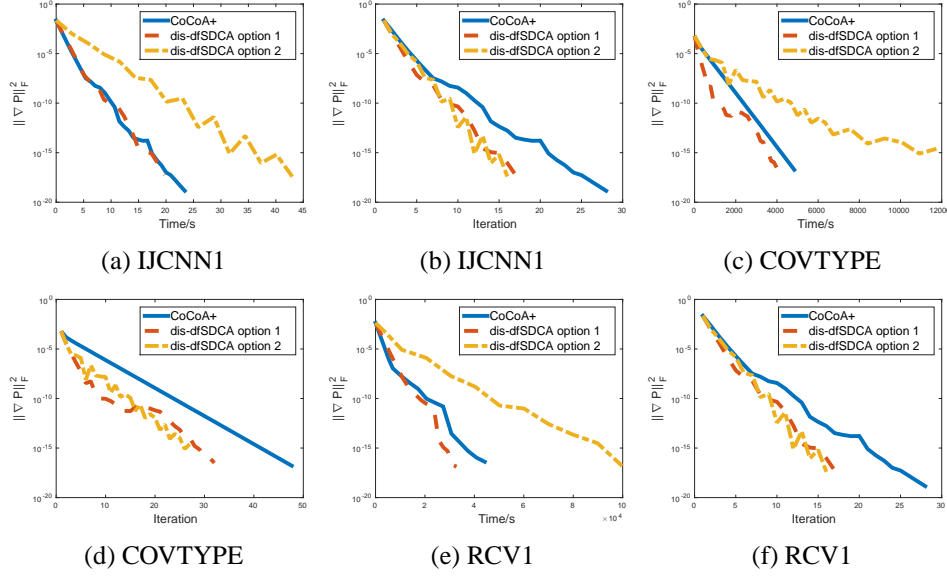


Figure 2: Experimental results of three compared methods. First column shows the convergence of $\|\nabla P\|^2$ with respect to time, and second column shows the convergence of $\|\nabla P\|^2$ with respect to iteration.

defined as:

$$\text{Time speedup} = \frac{\text{Running time for serial computation}}{\text{Running time of using } T \text{ workers}} \quad (23)$$

Both Figure 4a and 4b in Figure 4 illustrate that our method has near linear speedup when we increase the number of workers from $K = 1$ to $K = 16$. It is common that the speedup will degenerate when the number of worker becomes larger.

5.4 Impact of Maximum Time Delay Δ

As per the theoretical analysis in Theorem 1, maximum time delay Δ will affect the convergence rate of dis-dfSDCA. In this section, we perform experiment on IJCNN1 and COVTYPE datasets using dis-dfSDCA method with Option 1. In the experiment, we set $\lambda = 0.1$ for IJCNN1 dataset, and $\lambda = 0.3$ for COVTYPE dataset. Local inner iteration number $H = 1$ for both of these two experiments. Results of experiments are presented in Figure 3. We compare the convergence of duality gap and $\|\nabla P\|^2$ with respect to time or iteration. It is easy to notice that to achieve the same duality gap or ℓ_2 norm of full gradient, the number of iteration will go up when we increase the value of maximum time delay. This observation is compatible with our theoretical analysis in inequality (20). On the other hand, if the value of Δ becomes larger, the running time of our method will be smaller to achieve the same performance. It is for the reason that server does not need to waste time on waiting slow workers in the process.

6 Conclusion

In this paper, we propose a new Distributed Asynchronous Dual-Free Coordinate Ascent method, where bounded delay technique is used to control asynchrony. If ϕ_i is smooth and convex, using a constant learning rate, our method converges with a linear rate. We conduct experiments on Amazon Web Services with large-scale datasets. Experimental results show that dis-dfSDCA has better performance than state-of-the-art framework and demonstrate our theoretical analysis of its convergence property.

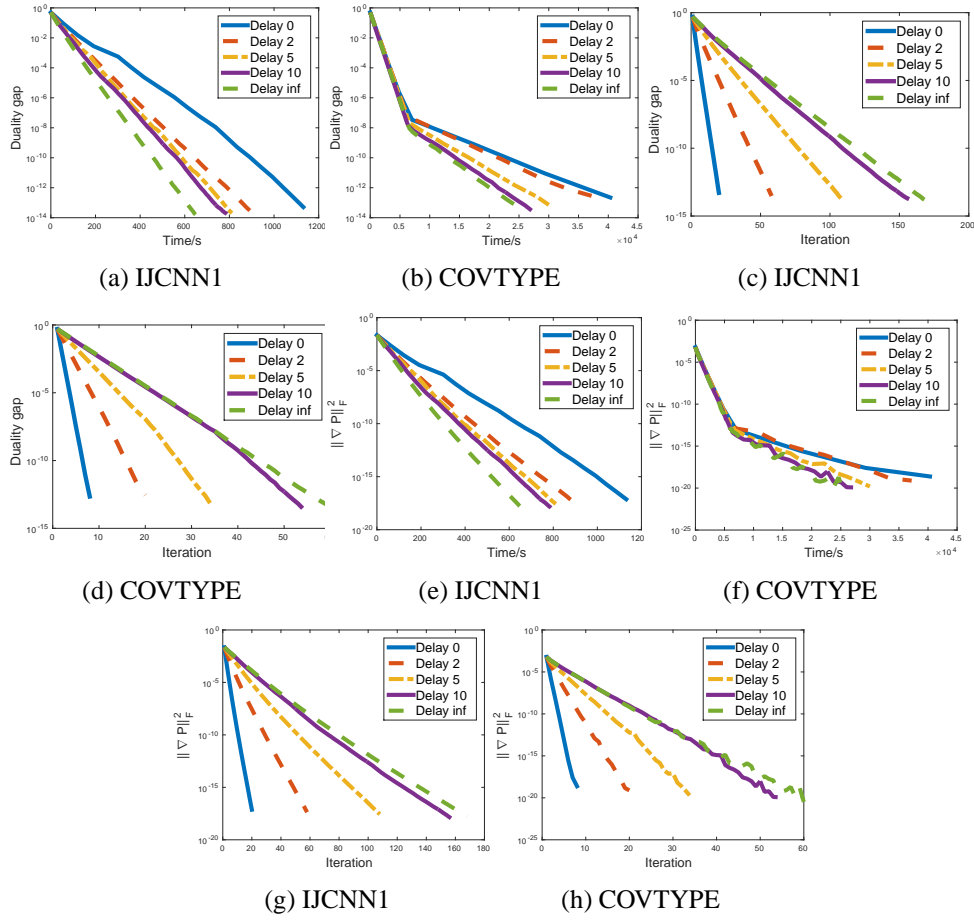


Figure 3: Experimental results of convergence rate with different maximum time delay Δ . First row shows the convergence of duality gap with respect to time and iteration, and second row shows the convergence of $\|\nabla P\|^2$.

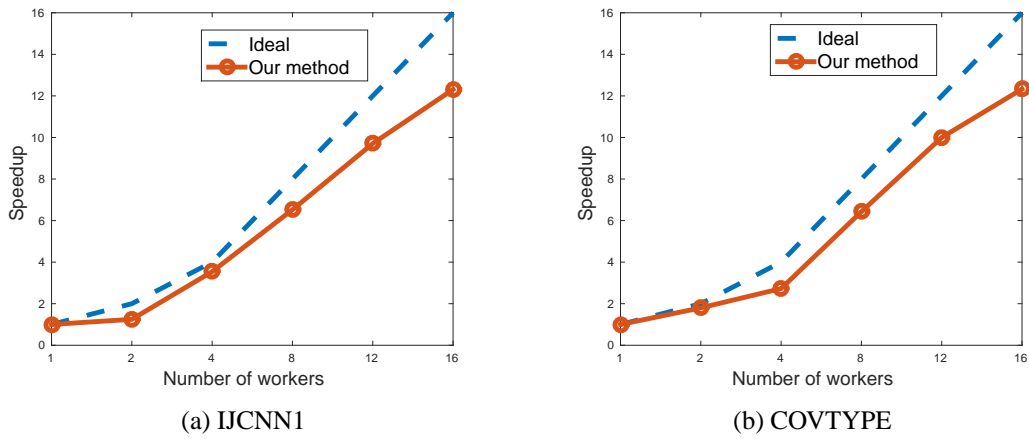


Figure 4: Running time speedup of our method on IJCNN1 and COVTYPE datasets. The dashed line denotes the ideal linear speedup.

References

- [1] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMP-STAT'2010*, pages 177–186. Springer, 2010.
- [2] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, Quoc V Le, et al. Large scale distributed deep networks. In *Advances in Neural Information Processing Systems*, pages 1223–1231, 2012.
- [3] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems*, pages 1646–1654, 2014.
- [4] Xi He and Martin Takáč. Dual free sdca for empirical risk minimization with adaptive probabilities. *arXiv preprint arXiv:1510.06684*, 2015.
- [5] Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S Sathya Keerthi, and Sellamanickam Sundararajan. A dual coordinate descent method for large-scale linear svm. In *Proceedings of the 25th international conference on Machine learning*, pages 408–415. ACM, 2008.
- [6] Zhouyuan Huo and Heng Huang. Asynchronous stochastic gradient descent with variance reduction for non-convex optimization. *arXiv preprint arXiv:1604.03584*, 2016.
- [7] Martin Jaggi, ETHZ CH, Michael I Jordan, BERKELEY EDU, Peter Richtárik, and Martin Takáč. Adding vs. averaging in distributed primal-dual optimization. 2015.
- [8] Martin Jaggi, Virginia Smith, Martin Takáč, Jonathan Terhorst, Sanjay Krishnan, Thomas Hofmann, and Michael I Jordan. Communication-efficient distributed dual coordinate ascent. In *Advances in Neural Information Processing Systems*, pages 3068–3076, 2014.
- [9] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, pages 315–323, 2013.
- [10] John Langford, Alexander Smola, and Martin Zinkevich. Slow learners are fast. *arXiv preprint arXiv:0911.0491*, 2009.
- [11] Mu Li, David G Andersen, Alex J Smola, and Kai Yu. Communication efficient distributed machine learning with the parameter server. In *Advances in Neural Information Processing Systems*, pages 19–27, 2014.
- [12] Xiangru Lian, Yijun Huang, Yuncheng Li, and Ji Liu. Asynchronous parallel stochastic gradient for nonconvex optimization. In *Advances in Neural Information Processing Systems*, pages 2719–2727, 2015.
- [13] Ji Liu, Stephen J Wright, and Srikrishna Sridhar. An asynchronous parallel randomized kaczmarz algorithm. *arXiv preprint arXiv:1401.4780*, 2014.
- [14] Yucheng Low, Danny Bickson, Joseph Gonzalez, Carlos Guestrin, Aapo Kyrola, and Joseph M Hellerstein. Distributed graphlab: a framework for machine learning and data mining in the cloud. *Proceedings of the VLDB Endowment*, 5(8):716–727, 2012.
- [15] Augustus Odena. Faster asynchronous sgd. *arXiv preprint arXiv:1601.04033*, 2016.

- [16] Zheng Qu and Peter Richtárik. Stochastic dual coordinate ascent with adaptive probabilities. 2015.
- [17] Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 693–701, 2011.
- [18] Sashank J Reddi, Ahmed Hefny, Suvrit Sra, Barnabás Póczos, and Alex J Smola. On variance reduction in stochastic gradient descent and its asynchronous variants. In *Advances in Neural Information Processing Systems*, pages 2629–2637, 2015.
- [19] Shai Shalev-Shwartz. Sdca without duality. *arXiv preprint arXiv:1502.06177*, 2015.
- [20] Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss. *The Journal of Machine Learning Research*, 14(1):567–599, 2013.
- [21] Martin Takác, Peter Richtárik, and Nathan Srebro. Distributed mini-batch sdca. *arXiv preprint arXiv:1507.08322*, 2015.
- [22] Lin Xiao and Tong Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.
- [23] Tianbao Yang. Trading computation for communication: Distributed stochastic dual coordinate ascent. In *Advances in Neural Information Processing Systems*, pages 629–637, 2013.
- [24] Ruiliang Zhang and James Kwok. Asynchronous distributed admm for consensus optimization. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1701–1709, 2014.
- [25] Ruiliang Zhang, Shuai Zheng, and James T Kwok. Fast distributed asynchronous sgd with variance reduction. *arXiv preprint arXiv:1508.01633*, 2015.
- [26] Tong Zhang and RUTGERS EDU. Stochastic optimization with importance sampling for regularized loss minimization. 2014.
- [27] Wei Zhang, Suyog Gupta, Xiangru Lian, and Ji Liu. Staleness-aware async-sgd for distributed deep learning. *arXiv preprint arXiv:1511.05950*, 2015.
- [28] Shen-Yi Zhao and Wu-Jun Li. Fast asynchronous parallel stochastic gradient descent: A lock-free approach with convergence guarantee. 2016.

A Proof of Lemma 1

Proof 1 Because $\alpha_i^* = -\nabla\phi_i(x_i^T w^*)$, we have the following inequality:

$$\begin{aligned}
& \sum_{t=1}^n \mathbb{E} \|\nabla\phi_i(w^{t-\tau}) + \alpha_i^* x_i\|^2 \\
&= \sum_{t=1}^n \mathbb{E} [\|\nabla\phi_i(x_i^T w^{t-\tau})x_i - \nabla\phi_i(x_i^T w^*)x_i\|^2] \\
&\leq 2 \sum_{t=1}^n \|\nabla\phi_i(w^{t-\tau}) - \nabla\phi_i(w^{t-1})\|^2 \\
&\quad + 2 \sum_{t=1}^n \|\nabla\phi_i(w^{t-1}) - \nabla\phi_i(w^*)\|^2 \\
&\leq 2L^2 \sum_{t=1}^n \|w^{t-1} - w^{t-\tau}\|^2 \\
&\quad + 2 \sum_{t=1}^n \|\nabla\phi_i(w^{t-1}) - \nabla\phi_i(w^*)\|^2 \\
&\leq 2L^2 \eta^2 \Delta^2 \sum_{t=1}^n \mathbb{E} \|v_i^t\|^2 + 4L \sum_{t=1}^n \left(P(w^{t-1}) - P(w^*) \right. \\
&\quad \left. - \frac{\lambda}{2} \mathbb{E} [\|w^{t-1} - w^*\|^2] \right)
\end{aligned} \tag{24}$$

where the first inequality follows from Lemma 3 in [18], and the second inequality follows from Lipschitz continuity of $\nabla\phi_i$, and the last inequality follows from the update rule of our algorithm and Lemma 1 in [19].

B Proof of Theorem 1

Proof 2 We split update of our algorithm into two stages, the inner loop is the traversal of all data, namely $t \in \{1, \dots, n\}$, and the outer loop $s \in \{1, \dots, S\}$ denotes the number of epoch. Then we know:

$$v_i^{s,t} = \left(\nabla\phi_i(x_i^T w^{s,t-\tau}) + \alpha_i^{s,t-\tau} \right) x_i \tag{25}$$

where, $\mathbb{E}[v_i^{s,t}] = \nabla P(w^{s,t-\tau})$. In epoch s , we use w^t to denote $w^{s,t}$, α_i^t to denote $\alpha_i^{s,t}$. Following the proof in [19], we define A_t and B_t :

$$A_t = \frac{1}{n} \sum_{i=1}^n \|\alpha_i^t x_i - \alpha_i^* x_i\|^2 \tag{26}$$

$$B_t = \|w^t - w^*\|^2 \tag{27}$$

Let $\beta = \eta\lambda n$, so in iteration t , $\alpha_i^t = (1 - \beta)\alpha_i^{t-1} + \beta(-\nabla\phi_i(x_i^T w^{t-\tau}))$. Then we know:

$$\begin{aligned}
& A_t - A_{t-1} \\
&= \frac{1}{n} \|\alpha_i^t x_i - \alpha_i^* x_i\|^2 - \frac{1}{n} \|\alpha_i^{t-1} x_i - \alpha_i^* x_i\|^2 \\
&= \frac{1}{n} \|(1 - \beta)(\alpha_i^{t-1} - \alpha_i^*)x_i + \beta(-\nabla\phi_i(x_i^T w^{t-\tau}) - \alpha_i^*)x_i\|^2 \\
&\quad - \frac{1}{n} \|\alpha_i^{t-1} x_i - \alpha_i^* x_i\|^2 \\
&= \frac{1}{n} \left((1 - \beta) \|\alpha_i^{t-1} x_i - \alpha_i^* x_i\|^2 + \beta \|\nabla\phi_i(x_i^T w^{t-\tau})x_i + \alpha_i^* x_i\|^2 \right. \\
&\quad \left. - \beta(1 - \beta) \|\alpha_i^{t-1} x_i + \nabla\phi_i(x_i^T w^{t-\tau})x_i\|^2 - \|\alpha_i^{t-1} x_i - \alpha_i^* x_i\|^2 \right) \\
&= \frac{\beta}{n} \left(-\|\alpha_i^{t-1} x_i - \alpha_i^* x_i\|^2 + \|\nabla\phi_i(x_i^T w^{t-\tau})x_i + \alpha_i^* x_i\|^2 \right. \\
&\quad \left. - (1 - \beta) \|v_i^t\|^2 \right) \\
&= \eta\lambda \left(-\|\alpha_i^{t-1} x_i - \alpha_i^* x_i\|^2 + \|\nabla\phi_i(x_i^T w^{t-\tau})x_i + \alpha_i^* x_i\|^2 \right. \\
&\quad \left. - (1 - \beta) \|v_i^t\|^2 \right)
\end{aligned} \tag{28}$$

In addition, it also follows that:

$$\begin{aligned}
& B_t - B_{t-1} \\
&= \|w^t - w^*\|^2 - \|w^{t-1} - w^*\|^2 \\
&= \|w^{t-1} - \eta v_i^t - w^*\|^2 - \|w^{t-1} - w^*\|^2 \\
&= -2\eta(w^{t-1} - w^*)^T v_i^t + \eta^2 \|v_i^t\|^2
\end{aligned} \tag{29}$$

We can also derive the following lower bound:

$$\begin{aligned}
& \mathbb{E}[(w^{t-1} - w^*)^T v_i^t] \\
&= \mathbb{E}[(w^{t-1} - w^{t-\tau} + w^{t-\tau} - w^*)^T v_i^t] \\
&= \eta \sum_{j=t-\tau+1}^{t-1} \mathbb{E}[(v_i^j)^T v_i^t] + (w^{t-\tau} - w^*)^T \nabla P(w^{t-\tau}) \\
&\geq \eta \sum_{j=t-\tau+1}^{t-1} \mathbb{E}[(v_i^j)^T v_i^t] + P(w^{t-\tau}) - P(w^*)
\end{aligned} \tag{30}$$

where the final inequality follows from convexity of $P(w)$.

Let's define $C_t = c_a A_t + c_b B_t$, and set $c_a = \frac{1}{2\lambda L}, c_b = 1$. Summing up $\mathbb{E}[C_t]$ from $t = 1$, and input

inequality (28) and (29) in this function, we have:

$$\begin{aligned}
& \sum_{t=1}^n \mathbb{E}[C_t] \\
= & \sum_{t=1}^n \mathbb{E} \left[c_a(1 - \eta\lambda)A_{t-1} + c_a\eta\lambda \|\nabla\phi_i(w^{t-\tau})x_i + \alpha_i^*x_i\|^2 \right. \\
& - c_a\eta\lambda(1 - \beta)\|v_i^t\|^2 + c_bB_{t-1} \\
& \left. - 2c_b\eta(w^{t-1} - w^*)^T v_i^t + c_b\eta^2\|v_i^t\|^2 \right] \\
\leq & \sum_{t=1}^n c_a(1 - \eta\lambda)\mathbb{E}[A_{t-1}] + \sum_{t=1}^n (c_b - 2c_a\eta L\lambda^2)\mathbb{E}[B_{t-1}] \\
& + \sum_{t=1}^n (c_b\eta^2 - c_a\eta\lambda(1 - \beta) + c_b\eta^2\Delta + 2c_a\lambda\Delta^2\eta^3L^2) \mathbb{E}[\|v_i^t\|^2] \\
& + \sum_{t=1}^n 4c_a\eta\lambda L(P(w^{t-1}) - P(w^*)) \\
& - \sum_{t=1}^n 2c_b\eta(P(w^{t-\tau}) - P(w^*)) \\
& + \sum_{t=1}^n c_b\eta^2 \sum_{j=t-\tau+1}^{t-1} \mathbb{E}[\|v_i^j\|^2] \\
\leq & \sum_{t=1}^n (1 - \eta\lambda)\mathbb{E}[C_{t-1}] + \sum_{t=1}^n \left(c_b\eta^2 - c_a\eta\lambda(1 - \beta) \right. \\
& \left. + 2c_b\Delta\eta^2 + 2c_a\lambda\Delta^2\eta^2L \right) \mathbb{E}[\|v_i^t\|^2] \\
\leq & \sum_{t=1}^n (1 - \eta\lambda)\mathbb{E}[C_{t-1}] \tag{31}
\end{aligned}$$

where the first inequality follows from Lemma 1 and inequality (30), input the value of c_a and c_b and let $\eta L \leq 1$, we have the second inequality, and the last inequality holds if:

$$c_b\eta^2 - c_a\eta\lambda(1 - \beta) + 2c_b\Delta\eta^2 + 2c_a\lambda\Delta^2\eta^2L \leq 0 \tag{32}$$

Therefore, if $\eta \leq \frac{1}{2L+n\lambda+4L\Delta+2\Delta^2L}$, we have:

$$\begin{aligned}
\sum_{t=1}^n \mathbb{E}[C_t] & \leq (1 - \eta\lambda) \sum_{t=1}^n \mathbb{E}[C_{t-1}] \\
& \leq \sum_{t=2}^n \mathbb{E}[C_{t-1}] + (1 - \eta\lambda)\mathbb{E}[C_0] \tag{33}
\end{aligned}$$

Thus, the inequality holds:

$$\mathbb{E}[C_n] \leq (1 - \eta\lambda)\mathbb{E}[C_0] \tag{34}$$

Because $\mathbb{E}[C_n] = \mathbb{E}[C_{s+1,0}]$ and $\mathbb{E}[C_0] = \mathbb{E}[C_{s,0}]$, we have:

$$\begin{aligned}\mathbb{E}[C_{s,0}] &\leq (1 - \eta\lambda)\mathbb{E}[C_{s-1,0}] \\ &\leq (1 - \eta\lambda)^s C_{0,0} \\ &\leq e^{-\eta\lambda s} C_{0,0}\end{aligned}\tag{35}$$

Above all, we have the final conclusion:

$$\begin{aligned}&\left[\|w^{s,0} - w^*\|^2 + \frac{1}{2n\lambda L} \sum_{i=1}^n \|\alpha_i^{s,0} x_i - \alpha_i^* x_i\|^2 \right] \\ &\leq e^{-\eta\lambda s} \left[\|w^{0,0} - w^*\|^2 + \frac{1}{2n\lambda L} \sum_{i=1}^n \|\alpha_i^{0,0} x_i - \alpha_i^* x_i\|^2 \right]\end{aligned}\tag{36}$$

C Proof of Theorem 2

Proof 3 Because the smoothness of P , we have this inequality: $P(w^{s,0}) - P(w^*) \leq \frac{L}{2} \|w^{s,0} - w^*\|^2$. From Theorem 1, we have:

$$P(w^{s,0}) - P(w^*) \leq C e^{-\eta\lambda s}\tag{37}$$

where C denotes a large constant. If $\eta = \frac{1}{2L+n\lambda+4L\Delta+2\Delta^2L}$, and $P(w^{s,0}) - P(w^*) \leq \varepsilon$ we have:

$$T \geq O\left(\left(\frac{2L + 4L\Delta + 2\Delta^2L}{\lambda} + n\right) \log\left(\frac{1}{\varepsilon}\right)\right)\tag{38}$$

